

# Data Visualisation - GGPlot2

*Webbinar - corona edition - Thomas Lin Pedersen from R*

Grammar of Graphics (ggplot2)

## Data import

readr - reading tabulated data

readxl - reading data from excel files

haven - reading data from spss

## Data Manipulation

tidyr - tidyverse

dplyr - tidyverse

data.table

## The Grammar of Graphics

Data is not just data

Representation defines what can be done with it

Grammar requires a tidy format

**Data:** Obviously

**Aesthetic Mappings:** this column represents x-axis, y-axis, etc

**Facet** mapping: link variables in the data to panels in the facet layout

**Statistics:** Applying some sort of transformation to the data to be able to plot it correctly, ex. Bar graphs need specified count, so data must contain that count.

**Scales:** graphical representable property, color, shapes, ex. mercedes -> blue, toyota -> red, etc.

Scales takes into consideration the different types of data

**Geometries:** how to specify how to interpret the data, ex. geo\_point will plot as a point while geo\_line will plot as a line (connect the points)

**Facets:** Splitting the data into different subplots, small multiples of the same data.

**Coordinates:** Defines the physical mapping of the aesthetics to the paper.

**Theme:** color of the background, style, pure style.

## Importing & Loading with Pacman

```
pacman::p_load(pacman, dplyr, GGally, ggplot2, ggrepel, patchwork, gifski, ggforce, ggthemes, maps, sf)
```

## Simple Scatter in GGPlot

Using the example data `faithful`, the simplest arguments for making a plot are the data, the columns (aesthetics), and the way to read it (geometries)

```
head(faithful)
```

```
ggplot(faithful, aes(eruptions, waiting)) +  
  geom_point(color='red')  
  
ggplot(faithful) +  
  geom_point(aes(eruptions, waiting, colour = eruptions < 3))  
  
ggplot(faithful) +  
  geom_point(aes(eruptions, waiting), colour = "steelblue")
```

## Histogram

```
ggplot(faithful) +  
  geom_histogram(aes(eruptions))
```

## combo

```
ggplot(faithful) + aes(eruptions, waiting) +  
  geom_point() +  
  geom_density2d()
```

## Exercises

```
ggplot(faithful) +  
  geom_point(aes(eruptions, waiting), shape = 'square', alpha = 0.6)  
  
ggplot(faithful) +  
  geom_histogram(aes(eruptions, colour = eruptions < 3.1))  
  
ggplot(faithful) +  
  geom_histogram(aes(eruptions, fill = eruptions < 3.1))  
  
ggplot(faithful) +  
  geom_histogram(aes(eruptions, fill = waiting < 60)) #Hard to see, but they're stacked together  
  
ggplot(faithful) +  
  geom_histogram(aes(eruptions, fill = waiting < 60), position = 'identity', alpha = 0.7) #Position adjusted  
  
ggplot(faithful) +  
  geom_point(aes(eruptions, waiting)) +  
  geom_abline(slope = -40, intercept = 200)
```

## Stats

Here `geom_bar()` uses `stat_count()` by default to count all the things inside the data. This is all done in the computer, so the computer loads the data and manipulates it. This is not feasible for enormous amounts of data, so its better to tell the data base to do the calculation and save it on a different data base.

```
ggplot(mpg) +
  geom_bar(aes(class))

mpg_counted <- mpg %>%
  count(class, name = 'count')

ggplot(mpg_counted) +
  geom_bar(aes(class, count), stat = 'identity')

ggplot(mpg_counted) +
  geom_col(aes(class, count)) #geom_col() is a shortcut to avoid putting 'identity' manually
```

Using stats:

```
ggplot(mpg) +
  geom_jitter(aes(class, hwy), width = 0.2) +
  stat_summary(aes(class, hwy), fun = mean, geom = 'point', colour = 'red', size = 5) # adding a point
```

Scales:

```
ggplot(mpg) +
  geom_point(aes(displ, hwy, colour = class))

ggplot(mpg) +
  geom_point(aes(displ, hwy, colour = class)) +
  scale_colour_brewer(name = 'CLASS', type = 'qual') #changing the name of the legend and inputing manually
```

```
ggplot(mpg) +
  geom_point(aes(displ, hwy)) +
  scale_x_continuous(breaks = c(3,5,6))+ #breaks means which numbers you want displayed in the axis.
  scale_y_continuous(trans = 'log10') # trans = transformation
```

*#all the display colours*

*#RColorBrewer::display.brewer.all()*

```
ggplot(mpg) +
  geom_point(aes(displ, hwy, colour = class, size = cyl)) +
  scale_color_brewer(type = 'qual') +
  scale_size_area(breaks = c(4,5,6,8)) #Size is also mapped to a continuous variable only 4,5,6,8
```

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy, colour = class, size = cty))
```

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy, colour = cty, size = cty))
```

```
ggplot(mpg) +
```

```
geom_point(aes(x = displ, y = hwy, colour = cty, size = cty)) +
guides(colour = 'legend') #only merges guides if they're exactly the same!
```

Facets: splitting the data under the same plot, subplots essentially.

```
ggplot(mpg) +
  geom_point(aes(displ, hwy)) +
  facet_wrap(~ class) #single variable
```

```
ggplot(mpg) +
  geom_point(aes(displ, hwy)) +
  facet_grid(year ~ drv) # two variables
```

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)) + # same scale for all
  facet_wrap(~ drv)
```

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)) + #different scales for all
  facet_wrap(~ drv, scales = 'free')
```

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)) + #only different y scales but same x scale
  facet_wrap(~ drv, scales = 'free_y')
```

```
ggplot(mpg) +
  geom_bar(aes(y = manufacturer)) +
  facet_grid(class ~ .)
```

```
ggplot(mpg) +
  geom_bar(aes(y = manufacturer)) +
  facet_grid(class ~ ., scale = 'free_y')
```

```
ggplot(mpg) +
  geom_bar(aes(y = manufacturer)) +
  facet_grid(class ~ ., space = 'free_y', scale = 'free_y')
```

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)) +
  facet_grid(year ~ drv)
```

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)) + #Less affective way of using it but sure here it is
  facet_wrap(~ year + drv)
```

## COORDINATES

```
ggplot(mpg) +
  geom_bar(aes(class)) +
  coord_polar()
```

```

ggplot(mpg) +
  geom_bar(aes(class)) +
  coord_polar(theta = 'y') +
  expand_limits(y = 70)

ggplot(mpg) +
  geom_bar(aes(class)) +
  scale_y_continuous(limits = c(0,40)) #you're trying to zoom in to your plot but if you try to set the

ggplot(mpg) +
  geom_bar(aes(class)) + #that is why you set the limits in the coordinate system instead, doesnt mess
  coord_cartesian(ylim = c(0,40))

```

```

ggplot(mpg) +
  geom_point(aes(hwy, displ)) +
  scale_y_continuous(trans = 'log') #breaks are horrible, because i'm doing it in scale

ggplot(mpg) +
  geom_point(aes(hwy, displ)) +
  coord_trans(y = 'log') #This only stretches the fabric after the calculations have been done to crea

```

## THEME

```

ggplot(mpg) +
  geom_bar(aes(y = class)) +
  facet_wrap(~year) +
  theme_minimal()

ggplot(mpg) +
  geom_bar(aes(y = class)) +
  facet_wrap(~year) +
  labs(title = "Number of car models per class",
       caption = "source: http://fueleconomy.gov",
       x = NULL,
       y = NULL)

```